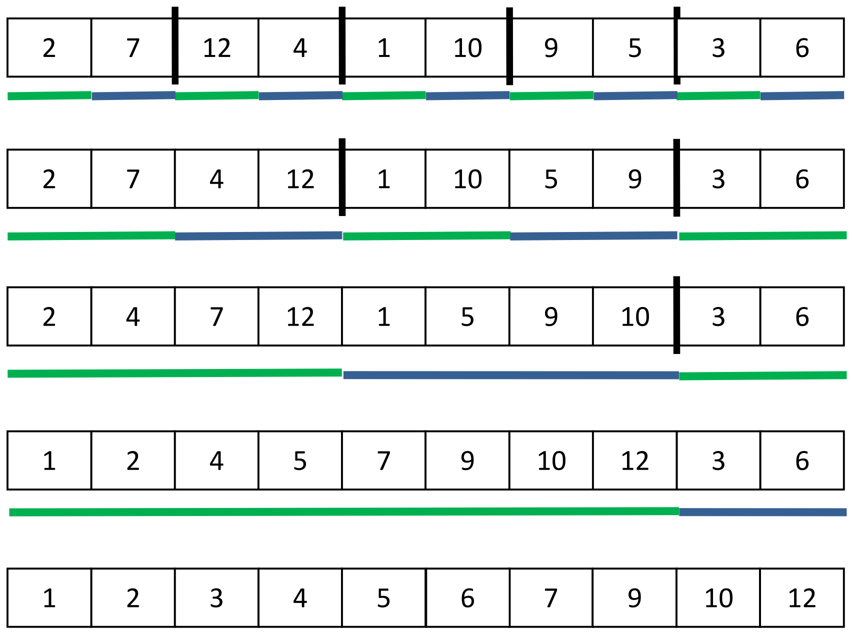
ZPR PWr – Zintegrowany Program Rozwoju Politechniki Wrocławskiej

Laboratory – List 6, continue Laboratory 5

Introduction

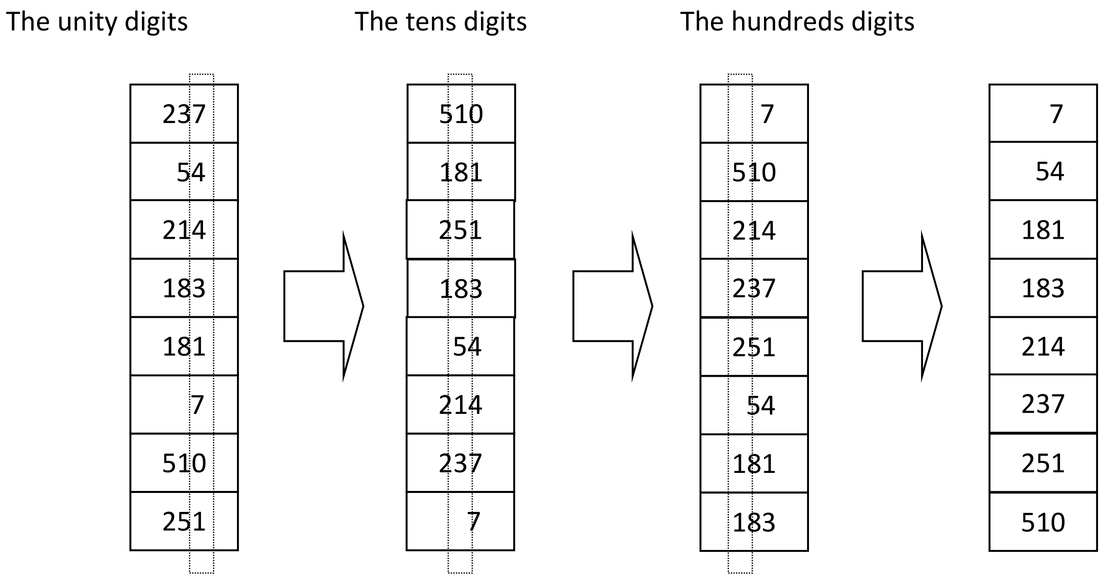
Iterative mergesort

Iterative mergesort is a version of merge sort, where you have to start from merging 1-element arrays into 2-elements arrays. Then you have to merge two 2-elements arrays into 4-elements arrays. Then 4-elements arrays into 8-elements arrays etc. Of course on the end of input array in can be less elements than expected. In such a case you have to merge arrays of smaller size. The example of such a sorting is presented below:



Radix sort

Radix sort is a sorting procedure which base on the digit’s position of integer value’s representation in a positional system One step in this sorting can be viewed as sorting using only one position of digits. In the first step it is used the less important digit, then the next digit until the most important digit. This is very important, that in the inner sorting it has to be used only stable sorting. An example is presented below:



To have the radix sort efficient it is needed to use a counting sort algorithm (presented on the lecture) as inner algorithm. "Derecognition" of a specific digit should be done without an additional table for this purpose, but by using an appropriate composition function of integer division and modulo operation.

Task list

1. Implement function `void iterativeMergeSort()` which sort (using iterative version of mergesort) in **increasing** order. Print state of the array (using `showArray`) before you start sort it and after each execution of outer loop.
2. Implement function **`void`** `radixSort()` which sort (using radix sort and counting sort) in **increasing** order. Let assume that weights will be from 1 to 999. Print state of the array (using `showArray`) before you start sort it and after each execution of outer loop.

For 100 points present solutions for this list till Week 8.
For 80 points present solutions for this list till Week 9.
For 50 points present solutions for this list till Week 10.
After Week 10 the list is closed.
There is a next page…

Appendix

It have to work all operation from previous list but additionally the following:

If a line has a format:
`mergesort`
`Main` program will call `iterativeMergeSort()` for current document.

If a line has a format:
`radixsort`
`Main` program will call `radixSort()` for current document.

The example from previous task is also proper for this list of tasks. A simple test for sorting from current list:

```
#Test for Lab6
go 10
ld doc1
link=c(10) and link=b(7)
write also link=z end finish link=a(20)
eod
add f(8)
add g(4)
add e(3)
show
mergesort
radixsort
ha
```

The output have to be:

```
START
!go 10
!ld doc1
!add f(8)
true
!add g(4)
true
!add e(3)
true
!show
Document: doc1
a(20) b(7) c(10) e(3) f(8) g(4) z(1)
! mergesort
20 7 10 3 8 4 1
7 20 3 10 4 8 1
3 7 10 20 1 4 8
1 3 4 7 8 10 20
! radixsort
20 7 10 3 8 4 1
20 10 1 3 4 7 8
1 3 4 7 8 10 20
1 3 4 7 8 10 20
!ha
END OF EXECUTION
```